Etape 1

- Manipuler des fichiers et répertoires avec IO : la classe File

Les opérations de manipulation de fichiers se font au moyen de la classe ***java.io.File***.

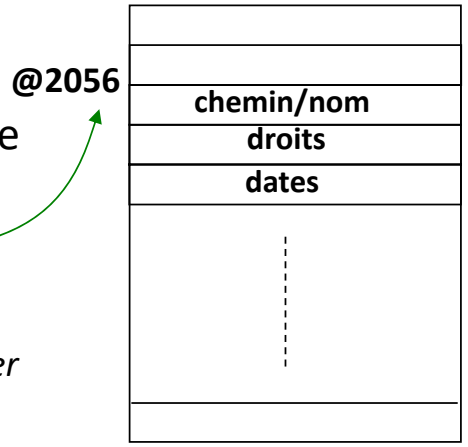
- La classe ***File*** est une représentation abstraite de tout fichier (fichiers ou répertoires).

  An abstract representation of file and directory pathnames

- Cette représentation est dite « abstraite » car le nom affecté ne correspond pas forcément à un fichier physique et désigne des fichiers ou des dossiers.

*File f = **new** File("chemin/nom");*

// à ce stade le File peut encore désigner un fichier ou un dossier



- Les méthodes de *File* permettent de *renommer un fichier, de le supprimer, d'en connaître les droits d'accès, …*

- Exemple

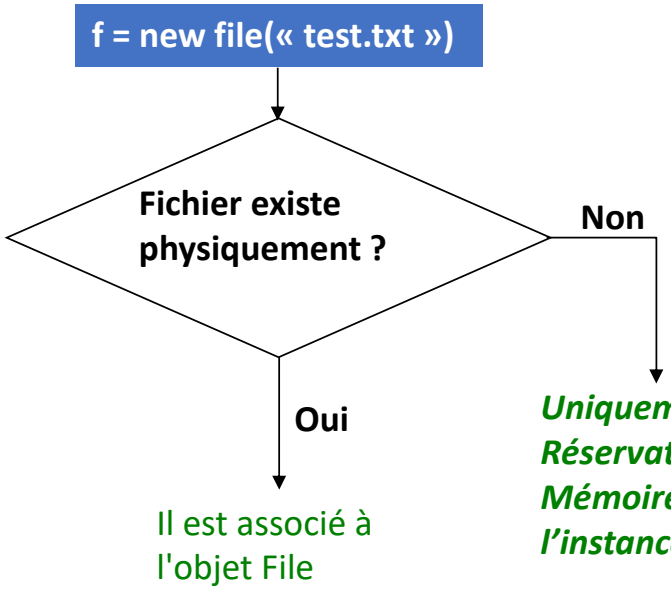
```
File f = new File("fichier.mp3"); // ne crée pas le fichier

System.out.println(f.getAbsolutePath()); // chemin complet
System.out.println(f.getName()); // nom seul

if (f.exists()) {
    System.out.println(f.getName() + " : " +
        (f.canRead()  ? "r" : "-") +
        (f.canWrite() ? "w" : "-") + " : " +
        f.length());
    f.delete();
}
```

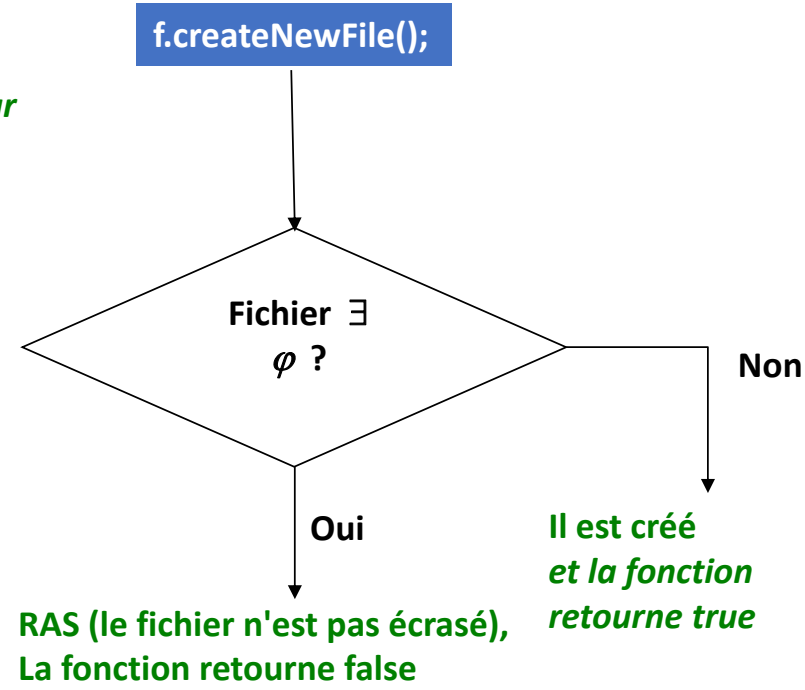
"exists : tests whether the file or directory denoted by this abstract pathname exists"

Instanciation d'un File et création d'un fichier



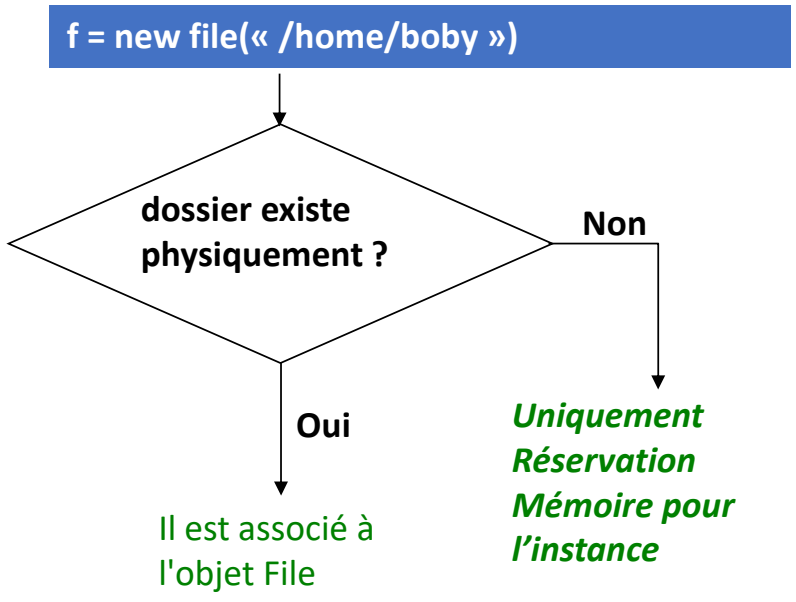
La taille, les droits et les autres propriétés sont celles du fichier physique

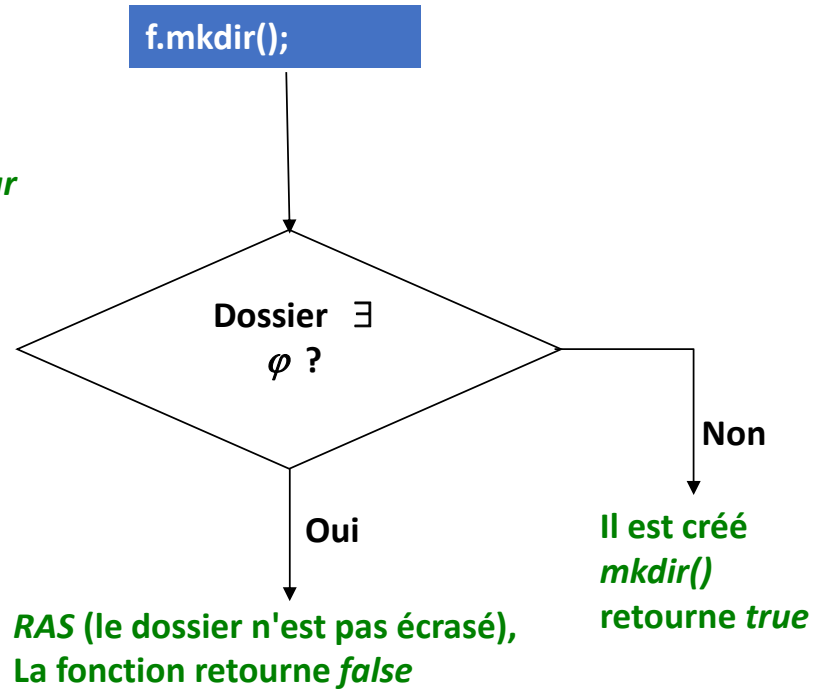
Création des files et fichiers/dossiers physiques



Instanciation d'un File et création d'un répertoire

Création des files et fichiers/dossiers physiques





Manipulation des fichiers

- Constructeurs :
  - *File(String pathname)* : ce constructeur prend en paramètre une chaîne de caractères (String) correspondant au chemin (relatif ou absolu) du fichier.
  - *File(String parent, String child) ou File(File parent, String child)* : paramètres le chemin (ou File) du répertoire parent et le nom du fichier.

- Donnée membre *static* :
  - *File.separator* : rend portable vos chemins : \ ou / selon les OS
  - *File.pathSeparator :* quand on a plusieurs chemins à spécifier ";" ou ":" selon les OS

- Méthodes :
  - *boolean canRead()* : permet de savoir s'il est possible lire le fichier.
  - *boolean canWrite()* : permet de savoir s'il est possible d'écrire dans le fichier.
  - *boolean createNewFile()* : permet de créer le fichier. Retourne *false* si ∃ déjà.
  - *boolean delete()* : supprime le fichier ou le chemin correspondant au nom. Si c'est un dossier => il doit être vide (sinon retourne *False*).
  - *boolean exists()* : teste si le File existe physiquement sur le disque.
  - *String getAbsolutPath()* : retourne chemin absolu du fichier.
  - *String getName()* : retourne le nom relatif avec l'extension.
  - *renameTo(File dest)* : renomme l'objet avec le file donné en paramètre.

Manipulation des chemins

Attribut static : *File.separator* ⇒ rend les chemins portable : \ ou / **selon les OS**

- static String **separator**

  The system-dependent default **name-separator character**, represented as a string for convenience. The system-dependent default name-separator character. This field is initialized to contain the first character of the value of the system property file.separator. On UNIX systems the value of this field is '/'; on Microsoft Windows systems it is '\\'.

- static char **separatorChar**

  The system-dependent default name-separator character.

Attribut static : *File.pathSeparator* ⇒ **si plusieurs chemins à spécifier** ";" ou ":" **selon les OS**

- static String **pathSeparator**

  The system-dependent **path-separator character**, represented as a string for convenience. This character is used to separate filenames in a sequence of files given as a *path list*. On UNIX systems, this character is ':'; on Microsoft Windows systems it is ';'.

- static char **pathSeparatorChar**

  idem en version caractère.

- *File getParentFile()* : retourne un File correspondant au répertoire parent.
- *boolean isDirectory()* : teste s'il s'agit d'un répertoire
- *boolean isFile()* : teste s'il s'agit d'un fichier.

Les deux méthodes peuvent avoir un paramètre un *FilenameFilter* pour filtrer les types de fichiers retournés

- *String[] list()* : le tableau retourné contient les noms des fichiers du répertoire.
- *File[] listFiles()* : retourne un tableau de File contenant les File représentants les fichiers du répertoire.
- *boolean mkdir()* : crée le répertoire représenté par le File.

Si on traite un fichier comme un répertoire on déclenche une exception.

```
if (f.isDirectory())
    {
    String [] files = f.list();
    for (int i =0; i < files.length; i++)
        {
            if (new File(args[0] + "\\" + files[i]).isDirectory())
                    { System.out.print("Rep : "); }
            else
                    { System.out.print("Fic : "); }

            System.out.println(args[0] + "\\" +   files[i]);
            }
     }
```